UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

PLATAFORMA DE ESPECIFICAÇÃO DE TESTES DE ACEITAÇÃO VIA INTERFACE DO UTILIZADOR

Mestrado em Informática

Versão Publica

Raquel Naré Maroco Martins Carmona

Trabalho de Projeto orientado por:

Prof. Doutor Francisco Cipriano da Cunha Martins

Eng. Joaquim Tiago Valério Pacheco Veríssimo dos Reis

2015

Agradecimentos

Em primeiro lugar, quero agradecer aos professores Francisco Martins e Eduardo Marques por todo o auxílio prestado e toda a colaboração e preocupação pelo sucesso deste projeto.

Em segundo lugar, gostaria de agradecer à empresa TFV – Sistemas Informáticos, S.A. pela oportunidade proporcionada.

Ao meu colega de projeto, Hugo Duarte, um obrigado pela sua ajuda, apoio e companheirismo.

Aos meus pais e avó, um agradecimento especial por me terem apoiado na realização da totalidade deste mestrado.

Ao meu amigo David, um muito obrigado pela sua paciência e perseverança comigo, apoio e força.

A minha amiga Debora agradeço todo o apoio e força durante o decorrer deste projeto.

A todos os meus amigos obrigada por compreenderem a minha ausência e por toda a vossa compreensão.

Aos meus pais e avó

Resumo

Num processo de engenharia de testes de *software*, os chamados testes de aceitação tem como objetivo a validação de uma aplicação de *soft*ware em relação aos requisitos do cliente. A execução destes testes envolve tipicamente a interação de um utilizador experiente com a aplicação em causa, por exemplo, através de ações ou validações sobre a interface da mesma.

Este dissertação descreve a criação de um sistema para a automatização de testes de aceitação sobre o Tripoint, um *software* para operadores turísticos e afins desenvolvido pela empresa TFV–Sistemas Informáticos, S.A. Atualmente, os testes de aceitação do Tripoint são levados a cabo pela equipa de *helpdesk* da empresa, processo esse que decorre com morosidade e baixo nível de automação. Estas deficiências tornam-se particularmente críticas num contexto de lançamento frequente de novas versões do Tripoint que ocorrem a cada 2 semanas, em média.

A ideia base do sistema de testes que propomos passa pela gravação da execução de um teste feito por um utilizador e a sua posterior reprodução automática. Durante a fase de gravação, pretende-se que o sistema registe as ações do utilizador sobre a interface gráfica do Tripoint e ainda asserções sobre o resultado dos testes configuradas pelo utilizador. A gravação produz a especificação de um caso de teste. Na fase de reprodução um teste deste tipo poderá ser executado sem intervenção humana, mediante injeção das ações sobre o interface gráfica e validação das asserções de teste.

Para a concretização deste projeto foram encontradas técnicas que permitissem a captura e reprodução de ações sob uma aplicação em ambiente Windows, bem como foi desenvolvida uma interface para o utilizador para facilitar a gravação e execução dos casos de teste.

Palavras-chave: Teste de aceitação, Execução de testes, Automação de testes.

Abstract

Acceptance test are a software testing process engineering aimed at validating a software application against end-user requirements. The execution of these tests typically involve the interaction of a domain expert with the application under test, for exemple, through actions or validations on the same interface.

This thesis describes a system for automated acceptance testing for Tripoint, called ARITEx, a software aplication for touristic and related enterprises by TFV– Sistemas Informáticos, S.A. Currently, Tripoint acceptance tests are conducted by the company's *helpdesk* team, resulting in a process that is slow, unrigorous and a low automates degree process. These disadvantages become particularly critical given that new versions of tripping are released frequently, every two weeks.

The basic idea of ARITEx is to record the execution of a test, and replay it later automatically. During the recording phase, the system records user-interface actions and user-specified test assertions, leading up to the recording of a test specification. During replay, a recorded test is automatically executed without human intervention through the injection of actions on the GUI and validation of test assertions.

For the realization of this project were found techniques that enable the capture and playback of shares in an application in Windows environment as well as an interface was developed for the user to facilitate the recording and execution of test cases.

Keywords: Acceptance Testing, Test Execution, Test automation.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Um teste de *software* exercita um sistema de *soft*ware ou componentes desse sistema, fornecendo-lhe dados de entrada e validando os seus resultados face ao comportamento esperado [1]. Globalmente, o objetivo da condução de testes de *software* é estabelecer com alto grau de confiança no uso do *software*, isto é, que este vai de encontro aos requisitos finais de utilização. Num processo maturo de engenharia de testes de *software*, a atividade de teste permeia todos os estágios de elaboração inicial de um sistema e a sua evolução ao longo do tempo.

Os testes de *software* podem ser considerados em vários níveis de granularidade dos seus componentes e/ou estágios de desenvolvimento, de forma complementar. Há testes a vários níveis, por exemplo: de validação face aos requisitos finais, os chamados testes de aceitação, de validação do *design* do sistema considerando os seus componentes todos integrados, ou testes sobre um módulo do sistema ou unidades de um módulo. Pelo que cada tipo de teste está relacionado diretamente com a fase de desenvolvimento do *software*. Os testes de aceitação devem ser realizados por utilizadores que tem um forte domínio do *software* a testar sendo os mesmo especializados nas funcionalidades deste *software* [1].

Um cenário típico para testar os vários níveis de testes e a forma como se relacionam com as atividades de desenvolvimento de *software* está representado na Figura 1.1 [1], também conhecido como o modelo em V. Este modelo isola cada uma das etapas de teste e a informação para cada nível é consequente da atividade de desenvolvimento associada.

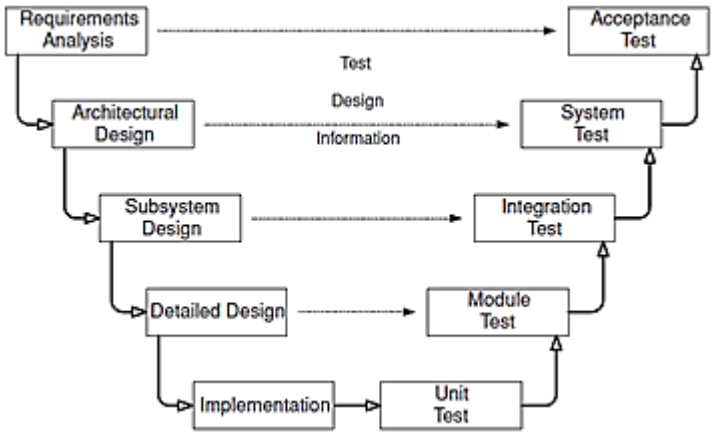

Figura 1.1 – Modelo V.

1.1 Motivação

Nos últimos anos a empresa TFV–Sistemas Informáticos, S.A desenvolveu o Tripoint [2], um *software* para a gestão global de operadores turísticos, agências de viagens e outros negócios relacionados com viagens. O Tripoint é um *software* de fácil utilização e compreensão, permitindo aos operadores turísticos e agências de viagem fazer toda a gestão de compra e venda de serviços relacionados com o turismo e um o controlo financeiro do negocio como um todo. Este *software* é utilizado principalmente em Portugal e no Brasil sendo instalado nas várias filiais de cada operador ou agência que adquire o Tripoint.

Entre diversas funcionalidades, o Tripoint permite pesquisar e reservar serviços como voos, alojamento, *transfers*, *rent-a-car*, pacotes turísticos, cruzeiros e outros serviços em tempo real, sendo ainda possível criar dinamicamente pacotes combinados de produtos próprios com produtos de terceiros. Os utilizadores podem ainda criar clientes e fornecedores, gerir a compra e a venda dos serviços, emitir faturas e fazer uma gestão financeira da respetiva filial. Em suma é um software de grandes dimensões e com casos de uso complexos.

Devido às várias solicitações dos clientes da TFV para acrescentar funcionalidades ao Tripoint, e corrigir erros encontrados pelos utilizadores, existe uma grande necessidade de gerar novas versões. Em cada uma das versões lançadas, os testes de aceitação são executados pela equipa de *helpdesk* da TFV. Pretende-se que este processo se torne mais sistemático, e em simultâneo libertar em maior grau a equipa de *helpdesk* para outras tarefas.

Com esta motivação, pretende-se construir um sistema em ambiente Windons para a automatização e validação dos testes de aceitação sobre o Tripoint.

1.2 Objetivos

Este trabalho tem como objetivo conceber e implementar um sistema de testes de aceitação automatizados, através da interação humana com a interface de utilizador do Tripoint. O sistema deve gerar os testes de aceitação de forma automática a partir das especificações (configurações e operações) escolhidas. As especificações deverão ficar registadas e poderão ser alvo de alterações futuras. A ideia é então criar um sistema com as seguintes funcionalidades:

- Captura de um teste mediante captura de ações e asserções especificadas pelo utilizador;
- Gravação da especificação do teste de forma persistente, por forma a poder ser executado posteriormente, ou editado programaticamente caso sejam necessárias alterações;
- Execução automática do teste, mediante injeção de ações gravadas na sessão do utilizador e validação de asserções.

1.3 Contribuições

As principais contribuições deste trabalho são:

- Desenho e implementação de um sistema (ARITEx) que permite a realização, edição e reprodução de testes de aceitação automática via interface do utilizador;
- Integração com a aplicação *Integration Test Runner* [3] no sentido da utilização dos testes produzidas pela ARITEx;
- Avaliação do sistema final através da:
    - Gravação e execução de testes de aceitação sobre o ARITEX;
    - Gravação e execução de testes de aceitação sobre o Tripoint;
    - Realização de testes de usabilidade com o utilizador final.

1.4 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 - Trabalho relacionado: analisa um conjunto de ferramentas de realização automatica de testes de aceitação;
- Capítulo 3 - A aplicação ARITEx: apresenta o *software* Tripoint e descreve a interface da aplicação desenvolvida e as suas funcionalidades;

- Capítulo 4 - Análise e desenho da solução: realiza uma análise do problema, e, mostra o desenho da solução;
- Capítulo 5 - Implementação da solução: descreve o processo de desenvolvimento, bem como as tecnologias usadas e o desenvolvimento de cada componente;
- Capítulo 6 - Avaliação: explica os testes realizados à aplicação e a sua importância para garantir a qualidade da ferramenta produzida;
- Capítulo 7 - Conclusões e trabalho futuro: apresenta as conclusões finais e possíveis desenvolvimentos futuros do trabalho realizado.

Capítulo 2

Trabalho relacionado

Neste capítulo serão apresentadas aplicações que permitem realizar testes de aceitação via interface de utilizador.

2.1 Coded UI Test

O Coded UI Test [4] é uma ferramenta de testes disponibilizada pelo Microsoft Visual Studio, que permite criar testes de interface de utilizador possibilitando, assim, testar se uma aplicação/*software* funciona corretamente.

Para este tipo de testes o Microsoft Visual Studio disponibiliza um construtor que surge minimizado no ecrã do utilizador que possui os seguintes botões: gravar/pausa/resumo do teste; editar passos; adicionar uma asserção e gerar o código fonte (ver Figura 2.1 [5]).

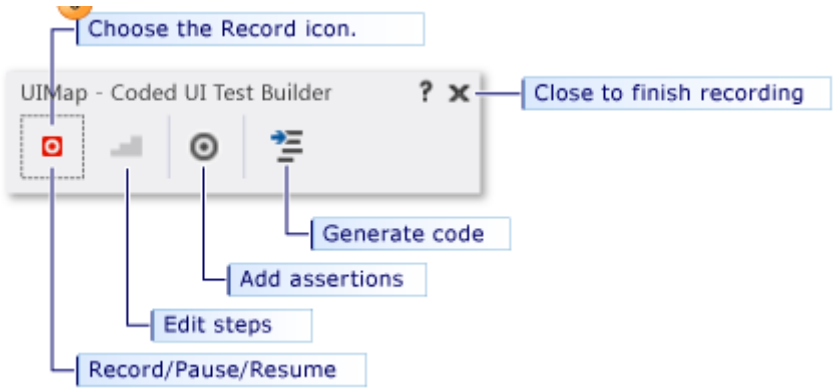

Figura 2.1 – Construtor de um teste do Coded UI Test

Assim quando pressionado o ícone de gravação o Coded UI Test inicia a deteção das ações do utilizador, possibilitando o mesmo interromper a gravação. As ações realizadas pelo utilizador apenas são editáveis durante a gravação e a única edição possível é a remoção de ações. Deste modo, o botão de editar mostra ao utilizador a lista de ações que este realizou que podem ser selecionadas e eleminadas.

As asserções aos elementos de uma aplicação só são possíveis após a conclusão da gravação das ações. Por fim, o Coded UI Teste gera o código C# ou visual basic para o respetivo teste, não sendo este editável por um utilizador não programador.

Cada teste desta ferramenta contêm um objeto UIMap que é específico para cada teste e representa as janelas, os controles, os parâmetros e as asserções. O código fonte de cada teste contem um comando que executa a ação realizada pelo utilizador. A sua reprodução vai verificar se os controlos, etc. exibidos são os corretos e se o seu valor também é correto [4].

Tendo em conta as funcionalidades descritas do Coded UI Test pretende-se aproveitar a sua grande maioria, entre elas: a gravação das ações do utilizador e a possibilidade de realizar asserções sobre os elementos do *software* em teste. No entanto o Coded UI Test não permite que um utilizador não programador altere um teste e que as asserções sejam realizadas no decorrer do destes.

Deste modo, pretende-se que a plataforma: detete e registe as ações realizadas sob a interface a testar, seja possível realizar as asserções durante a gravação das ações; o teste seja editável após a sua criação e por um utilizador não programador.

2.2 Selenium

O Selenium [6] é uma ferramenta de automação de testes para aplicações web que é constituída por um conjunto de ferramentas de *software* distintas e cada uma delas com funcionalidades diferentes para auxiliar a automação de testes. As principais ferramentas deste *software* são o Selenium IDE e RC.

O IDE é uma extensão do firefox que auxilia o utilizador na criação de casos de testes. Esta ferramenta grava as ações do utilizador sobre o browser gerando assim um caso de teste. Com esta ferramenta é possível realizar asserções sobre os elementos da página web a testar. É ainda possível realizar alterações a testes já gravados, como por exemplo, acrescentar novas ações; copiar, colar e excluir ações.

Como o auxílio da ferramenta Selenium RC e o IDE consegue reproduzir um teste ou um conjunto de testes, Figura 2.2 [6].

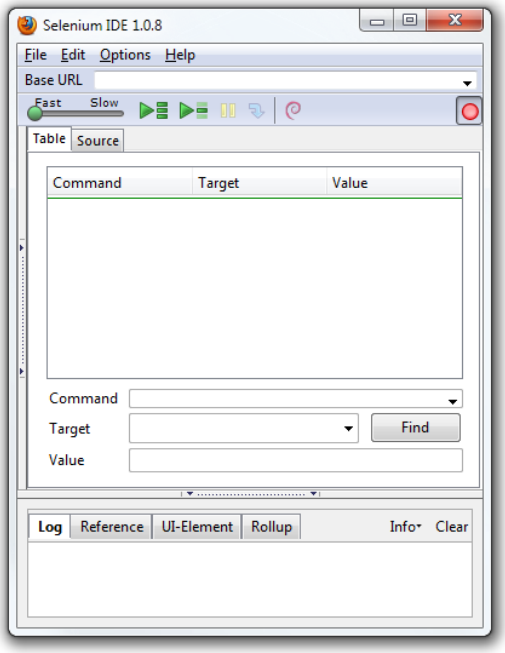

Figura 2.2 – Aspeto gráfico do Selenium IDE

As funcionalidades do Selenium são as pretendidas para a plataforma a criar. Uma vez que esta ferramenta só funciona para aplicações web pretende-se adaptar estas funcionalidades para o caso do projeto.

2.3 FitNesse

O FitNesse [7] é um wiki[1] web server onde é possível realizar testes a uma qualquer aplicação onde se testa as camadas de negócio da aplicação. Nesta ferramenta não é possível automatizar os testes. O FitNesse realiza testes de aceitação possuindo métodos para determinar se a aplicação em teste está a funcionar corretamente de uma forma automática.

A forma de criar testes no FitNesse é através de tabelas, em que são introduzidos nesta ferramenta os valores de entrada e os valores esperados e ela automaticamente faz a verificação dos resultados. Este tipo de testes realizados por esta ferramenta não são mais do que asserções à aplicação. Na Figura 2.3 [7] encontra-se um exemplo de teste para verificar se uma calculadora está a funcionar corretamente em relação à operação de divisão.

[1] Wiki é uma aplicação web que permite ao utilizador adicionar, alterar e remover conteúdos em colaboração com outros utilizadores.

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	~=3.14
9	3	<5
11	2	4<_<6
100	4	33

Figura 2.3 – Especificação do teste a uma calculadora à operação divisão

O modo como o FitNess específica as asserções para os utilizadores pode-se tonar interessante de integrar na plataforma a desenvolver uma vez que os utilizadores apenas tem que introduzir os valores de input e output.

2.4 EasyAccept

O EasyAccept [8] é uma ferramenta para executar testes de aceitação. Esta ferramenta interpreta um conjunto de script e executa-os. Cada um destes script tem que ser criado pelo utilizador. Esta ferramenta acede ao programa a ser testado através de um método de fachada que é introduzido em cada script de teste. Este vai correr cada um dos script introduzidos e avaliar os valores de saída ou comportamento da aplicação em teste e verifica se o comportamento foi o esperado.

No fim da execução de cada teste esta ferramenta mostra ao utilizador as divergências entre os resultados obtidos e os esperados.

Para a plataforma a criar esta ferramenta seria útil para executar as ações que geram um teste à interface.

2.5 Visual UI Automation Verify

O Visual UI Automation Verify (Visual UIA Verify) [9] é um driver do Windows GUI para a biblioteca UI Automation test que permite realizar testes de aceitação de forma manual para uma interface de utilizador.

A interface fornecida (Figura 2.4 [9]) utiliza a biblioteca em causa e permite eliminar o processo de codificação de linhas de código. Nesta aplicação quando o utilizador clica num elemento da sua aplicação de teste o Visual UIA Verify deteta toda a árvore do seu elemento, ou seja, os seus pais, ele próprio e os seus filhos. Para cada um dos elementos selecionados é possível realizar testes de forma automática. Cada um dos testes não é passível de gravação, logo não é possível reproduzi-lo. Esta aplicação poderá ser bastante útil para facilitar toda a árvore de um elemento.

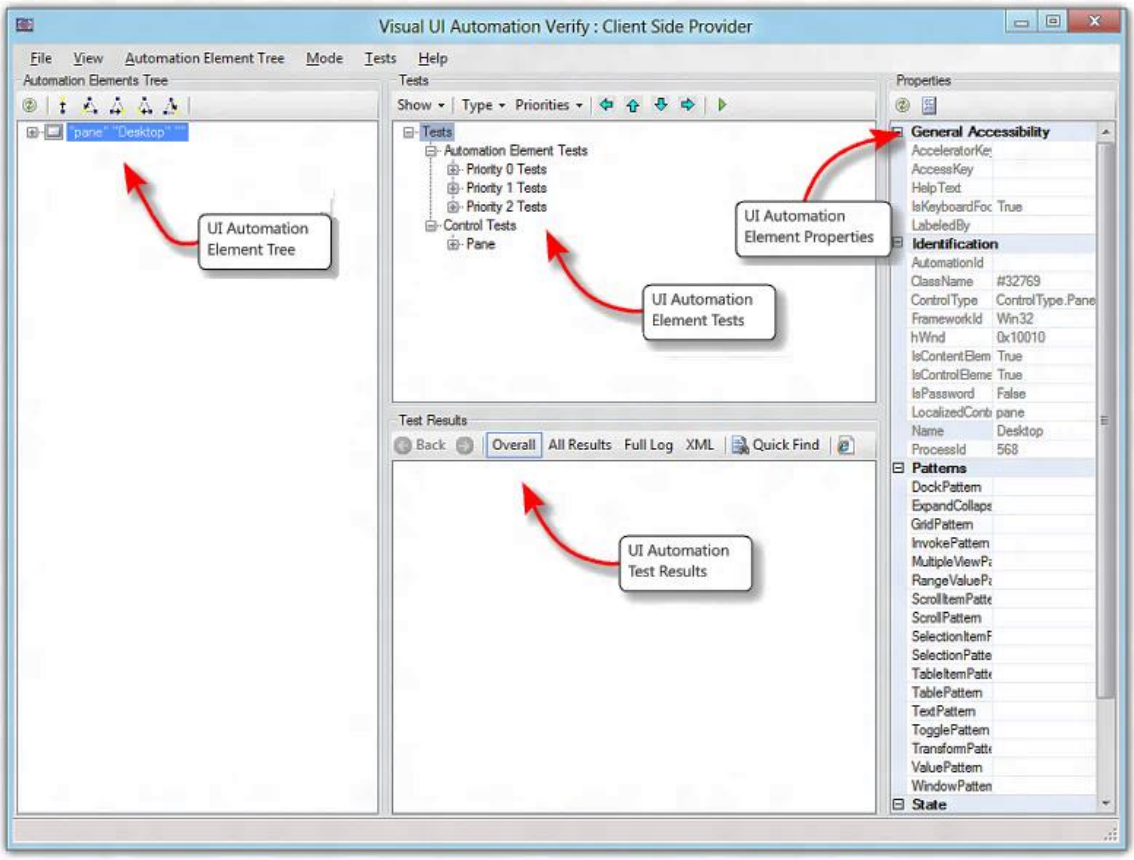

Figura 2.4 – Principais áreas funcionais da interface do utilizador do Visual UI Automation Verify

Capítulo 3

A aplicação ARITEx

Este capítulo apresenta-se a aplicação ARITEx em termos das suas funcionalidades gerais tendo em conta os objetivos traçados para a mesma. Os conceitos subjacentes à aplicação são expostos primeiramente, seguindo-se uma perspetiva geral das suas funcionalidades. Os principais itens de funcionalidades são a seguir descritos em detalhe compreendendo: janela de entrada para autenticação, janela principal da aplicação, criação, edição e reprodução de testes. Para compreender o contexto para a qual o ARITEx foi devolvidos apresenta-se em mais detalhe as funcionalidades do Tripoint bem como o seu uso e as validações que são realizadas durante a realização de testes de aceitação.

3.1 Enquadramento

3.1.1 Funcionalidades do Tripoint

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial a mesma expõem as funcionalidades de utilização do Tripoint.

3.1.2 Uso e validação do Tripoint

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem exemplos de uso do Tripoint e as respetivas validações realizadas a quando da realização de testes de aceitação sob o sistema.

3.2 Conceitos

Neste trabalho ir-se-á falar de algumas noções que poderão não ser claras para o leitor. De modo a clarificar o leitor foram identificados um conjunto de conceito que se pretende esclarecer.

A restante secção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial, a mesma expõe um conjunto de conceitos que são tomados no contexto deste projeto que são: passo de teste, teste, cenário e configuração.

3.3 Perspetiva geral do interface

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se ao leitor uma perspetiva geral da interface da ferramenta desenvolvida.

3.4 Autenticação

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o comportamento da janela de autenticação.

3.5 Janela principal

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o comportamento da janela principal da ferramenta.

3.6 Criação de testes

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o modo de criação de um teste.

3.7 Reprodução de testes

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o modo de reprodução de um teste.

Capítulo 4

Análise e desenho da solução

Este capítulo descreve a análise e desenho da aplicação ARITEx. Identificamos os requisitos funcionais e não-funcionais que se levaram em conta no desenvolvimento, descrevemos os casos de uso que capturam os requisitos funcionais mais relevantes, e apresentamos a arquitetura para a aplicação.

4.1 Requisitos

4.1.1 Requisitos funcionais

Os requisitos funcionais definem as funcionalidades do sistema e seus componentes. Ao longo de reuniões que tiveram lugar durante dois meses, estes foram identificados tendo em conta a natureza de uma ferramenta de captura e reprodução de testes e os aspetos específicos levantados pela TFV e operações do Tripoint.

A restante secção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial, a mesma expõe os requisitos funcionais da aplicação.

4.1.2 Requisitos não-funcionais

Os requisitos não-funcionais definem um conjunto de critérios para a operação do sistema na forma de restrições, fatores de qualidade de serviço e aspetos de arquitetura.

A restante secção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial, a mesma expõe os requisitos não-funcionais da aplicação.

4.2 Casos de uso

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial mostra-se o diagrama de casos de uso do sistema e que são especificados de seguida.

4.2.1 Criação de teste

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial especifica-se o caso de uso que diz respeito à criação de um caso de teste.

4.2.2 Reprodução de teste

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial especifica-se o caso de uso que diz respeito à reprodução de um caso de teste.

4.3 Arquitetura

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial descreve-se a arquitetura do sistema e da solução a implementar.

Capítulo 5

Implementação da solução

Este capítulo descreve o desenvolvimento da aplicação ARITEx. Expondo as várias ferramentas usadas na conceção da aplicação e o seu uso na estruturação da aplicação em camadas de dados, serviço e negócio.

5.1 Tecnologias usadas

5.1.1 WPF e MVVM

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial descreve-se a utilização do WPF e MVVM neste projeto.

5.1.2 Bibliotecas de captura e injeção de eventos

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial descreve-se a utilização das 3 bibliotecas utilizadas para a captura e injeção de eventos.

5.1.3 Motor de bases de dados

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial descreve-se a utilização dos motores de base de dados: ASP.NET Web API e Entity Framework.

5.2 Camada de dados

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o modelo relacional da base de dados.

5.3 Camada de serviços

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se toda a camada de serviços da aplicação.

5.4 Camada de negócio

5.4.1 Objetos de negócio

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se os objetos de negócio utilizados para a implementação.

5.4.2 Gravador

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o modo como se realiza a gravação das ações dos utilizadores.

5.4.3 Executor

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se o modo como se realiza a reprodução das ações dos utilizadores.

5.5 Camada de apresentação

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se como foi implementada a interface gráfica da aplicação.

Capítulo 6

Avaliação

Neste capítulo apresentamos uma avaliação realizada à aplicação ARITEx, com três vertentes:

1) Validação do ARITEx face aos seus casos de uso, usando a própria aplicação para a condução dos testes de sistema correspondentes, isto é, a ARITEx é usada para se testar a si mesma;
2) Aferição da eficácia do ARITEx na execução automática de um conjunto de 30 testes significativos sobre o Tripoint;
3) Avaliação empírica do ARITEx através de testes de usabilidade usando operadores humanos.

6.1 Validação do ARITEx

Após se ter concluído a implementação de todas as camadas foi necessário efetuar uma bateria de testes de modo a garantir que todo o sistema estava a funcionar corretamente. O objetivo destes testes é interagir com o sistema sob o ponto de vista do utilizador final, de modo a percorrer todas as funcionalidade na procura de alguma falha ou mau funcionamento do sistema.

A restante secção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial, a mesma expõe as validações realizadas ao ARITEx sendo para tal utilizado o próprio.

6.2 Execução de testes sobre o Tripoint

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se os tempos de gravação e execução de testes sobre o Tripoint usando o ARITEx.

6.3 Testes de usabilidade

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se os testes de usabilidade realizados à ferramenta desenvolvida.

6.3.1 Metodologia

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se a metodologia utilizada nos testes de usabilidade.

6.3.2 Resultados

Esta seção foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõem-se e interpreta-se os resultados obtidos com os testes de usabilidade.

Capítulo 7

Conclusões e trabalho futuro

Neste capítulo são apresentadas as considerações finais e as perspetivas futuras.

7.1 Conclusões

Este documento apresentou a aplicação ARITEx com todas as suas funcionalidades e modo de implementação. Após a conclusão do mesmo viram-se assim compridos os objetivos propostos para o projeto (ver secção 1.2). Assim é possível através do ARITEx gravar teste de aceitação via interface do utilizador, para a aplicação Tripoint ou outra aplicação Windows usando uma versão do ARITEx modificada, e sua posterior reprodução de uma forma automática.

Concluídas as fases de desenho e implementação do ARITEx foi feita uma avaliação da ferramenta compreendendo: validação dos casos de uso usando a própria ferramenta para autoavaliação, execução de testes sobre o Tripoint, e ainda testes de usabilidade com os potenciais utilizadores finais. Conclui-se de forma geral que a ferramenta cumpre o propósito para o qual foi concebida e que foi bem recebida pelos seus utilizadores.

Após a fase de avaliação, foram revistos alguns aspetos de utilização e corrigidos pequenos erros na aplicação, passando esta a estar disponível para utilização final.

7.2 Trabalho futuro

Como trabalho futuro, identificamos a seguir um conjunto de novas funcionalidades e melhorias a realizar no ARITEx. Com respeito a novas funcionalidades, consideramos os seguintes pontos:

- Habilitar mecanismos de inicialização e reposição da base de dados usada pelo Tripoint. Na versão atual, estas são tarefas manuais do utilizador antes ou após a execução dos testes. Para automação neste processo pode-se

recorrer por exemplo a scripts SQL que executam em integração com o ARITEx antes ou depois de um teste ou conjunto de testes.

- Generalizar o ARITEx por forma a validar uma aplicação do Windows arbitrária em vez de apenas o Tripoint. Atualmente isso é possível apenas usando uma versão modificada do ARITEx que não tem a generalidade desejada. Esta generalização pode ser conseguida realizando uma configuração inicial antes de se iniciar a gravação dos testes ou reprodução esta deteção pode ser conseguida através da conjugação de dois controlos sobre a janela inicial do programa que se pretende testar.
- Habilitar asserções sobre imagens no Tripoint (como as referidas na secção 113.1.2), que é incompatível com a framework do UI Automation. Ter-se ia que encontrar uma outra framework que permita a deteção e comparação de imagens.
- Habilitar asserções a propriedades específicas do texto como a cor ou a negrito. Esta propriedade do texto não faz parte das propriedades acessíveis do texto através do UI Automation. Ter-se ia que encontrar uma outra framework que permitisse realizar este tipo de asserções.

Os melhoramentos a realizar terão em conta os aspetos apontados como menos favoráveis no decorrer dos testes de usabilidade (descritos na Secção 6.3)

- As mensagens de erro e/ou de teste terminado com sucesso deverão surgir como *pop up* para o utilizador;
- Aumentar a rapidez na aquisição dos dados a partir da base de dados do ARITEx;
- Realizar um estudo para averiguar o modo de facilitar o uso da aplicação no que diz respeito ao conceito de asserção.

Bibliografia

[1] P. Ammann e J. Offutt, Introduction to software testing, New York: Cambridge University Press, 2008.

[2] "TFV, S.A.," [Online]. Available: http://www.tfv.pt/Home. [Acedido em Junho 2015].

[3] H. Duarte, "Plataforma de execução de testes de aceitação," Faculdade de Ciências da Universidade de Lisboa, 2015 (Em publicação).

[4] Microsoft, "How to: Create a Coded UI Test," [Online]. Available: http://msdn.microsoft.com/en-us/library/dd286681(v=vs.100).aspx. [Acedido em 25 Novembro 2014].

[5] Microsoft, "Verifying Code by Using UI Automation," [Online]. Available: http://msdn.microsoft.com/en-us/library/dd286726.aspx. [Acedido em 25 Novembro 2014].

[6] D. Burns, Selenium 1.0 Testing Tools: Beginner's Guide, Olton: Packt Publishing, 2010.

[7] G. Adzic, Test Diven .Net Development whit FitNess, London: Neuri Limited, 2009.

[8] J. P. Sauvé, O. L. Abath Neto e W. Cirne, "EasyAccept: A Tool to Easily Create, Run and Drive Development with Automated Acceptance Tests," em *AST'06*, Shanghai, 2006.

[9] Microsoft, "Visual UI Automation Verify," [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/jj160544(v=vs.85).aspx. [Acedido em 25 Novembro 2015].

[10] A. Cockburn, Writing effective use cases, Addison-Wesley, 2001.

[11] Microsoft, “Windows Presentation Foundation,” 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx. [Acedido em Junho 2015].

[12] Microsoft, “5: Implementing the MVVM Pattern Using the Prism Library 5.0 for WPF,” 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx. [Acedido em Junho 2015].

[13] Microsoft, “UI Automation Overview,” 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms747327(v=vs.110).aspx. [Acedido em Junho 2015].

[14] Microsoft, “C# Programming Guide,” 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx. [Acedido em Junho 2015].

[15] TestStack, “White Home,” 2013. [Online]. Available: http://teststack.azurewebsites.net/white/index.html. [Acedido em Março 2015].

Anexo A

Casos de uso

Este anexo foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõe-se os vários casos de uso da ferramenta a desenvolvida.

Anexo B

Diagrama de classes

Este anexo foi omitida devido à natureza confidencial deste projeto. Na versão confidencial expõe-se o diagrama classes da ferramenta a desenvolvida.

Anexo C

Questionário de usabilidade

Este anexo foi omitida devido à natureza confidencial deste projeto. Na versão confidencial mostra-se o questionário de usabilidade utilizado para os respetivos testes.